

Contents

1 Routine/Function Prologues	2
1.0.1 iniTimeVar.F90 (Source File: iniTimeVar.F90)	2

1 Routine/Function Prologues

1.0.1 iniTimeVar.F90 (Source File: iniTimeVar.F90)

Initialize the following time varying variables:

water : h2osno, h2ocan, h2osoi_liq, h2osoi_ice, h2osoi_vol

snow : snowdp, snowage, snl, dz, z, zi

temperature: t_soisno, t_veg, t_grnd

Note - h2osoi_vol is needed by clm_soilalb -this is not needed on restart since it is computed before the soil albedo computation is called

Note - remaining variables are initialized by calls to ecosystem dynamics and albedo subroutines.

Method: Initial data is saved to instantaneous initial data files for each of the [maxpatch] subgrid patches for each of the [numland] land points. If a subgrid patch is not active (e.g., 3 patches rather than [maxpatch]), the inactive subgrid patches have data values for the first subgrid patch. This way, as long as the land mask DOES NOT change among runs (i.e., [numland] is the same), an initial data file can be used in numerous experiments even if the surface types (and hence [numpatch]) differ

Author: Mariana Vertenstein

INTERFACE:

```
subroutine iniTimeVar (readini, eccen, obliqr, lambm0 , mvelpp, lis, tile)
```

USES:

```
use lis_module
use tile_module
use precision
use clm_varder
use clm_varctl, only : clmdrv
use clm_varmap , only : numpatch
use clm_varcon , only : bdsno, istice, istwet, istsoil, denice, denh2o, tfrz, spval
use inicFileMod , only : type_inidat, inicrd, histrd
use shr_sys_mod , only : shr_sys_abort
use spmdMod      , only : masterproc
use time_manager, only : get_nstep, get_curr_calday
#if (defined SPMD)
use mpishorthand, only : mpicom, mpichar
#endif
```

CONTENTS:

```
if (readini) then
  if ( masterproc ) write (6,*) 'Reading initial data '
  call type_inidat(initype)
  if (trim(initype) == 'INICFILE') then
    call inicrd ()
  else if (trim(initype) == 'HISTFILE') then
    call histrd ()
```

```

    else
        call shr_sys_abort('initial data type is limited to INIC or HIST file only')
    endif
    do k = 1,numpatch
        do j = 1,nlevsoi
            clm(k)%h2osoi_vol(j) = clm(k)%h2osoi_liq(j)/(clm(k)%dz(j)*denh2o) &
                + clm(k)%h2osoi_ice(j)/(clm(k)%dz(j)*denice)
        end do
    end do
else
    if ( masterproc ) write (6,*) 'Setting initial data to non-spun up values'

! =====
! Set snow water
! =====

! NOTE: h2ocan, h2osno, snowdp and snowage has valid values everywhere

do k = 1,numpatch
    if (lis%o%startcode == 4) then
        clm(k)%h2ocan = 0.
        clm(k)%snowage = 0.
    else
        clm(k)%h2ocan = 0.
        if (clm(k)%itypwat == istice) then
            clm(k)%h2osno = 1000.
        else
            clm(k)%h2osno = clmdrv%clm2_iscv
        endif
        clm(k)%snowdp = clm(k)%h2osno/bdsno
        clm(k)%snowage = 0.
    endif
end do

! =====
! Set snow layer number, depth and thickness
! =====

call snowdp2lev ()

! =====
! Set snow/soil temperature
! =====

! NOTE:
! t$-$soisno only has valid values over non-lake
! t$-$lake only has valid values over lake
! t$-$grnd has valid values over all land
! t$-$veg has valid values over all land

```

```

do k =1,numpatch
  clm(k)%t_soisno(-nlevsno+1:nlevsoi) = 0
  if (lis%o%startcode == 4) then
    clm(k)%t_veg = clm(k)%forc_t
  else
    clm(k)%t_veg = clmdrv%clm2_it
  endif
  if (.not. clm(k)%lakpoi) then !not lake
    clm(k)%t_soisno(-nlevsno+1:0) = spval
    if (clm(k)%snl < 0) then !snow layer temperatures
      do i = clm(k)%snl+1, 0
        if (lis%o%startcode == 4) then
          if (clm(k)%forc_t < 273.16) then
            clm(k)%t_soisno(i) = clm(k)%forc_t
          else
            clm(k)%t_soisno(i) = 273.16 - 1.
          endif
        else
          if (clmdrv%clm2_it < 273.16) then
            clm(k)%t_soisno(i) = clmdrv%clm2_it
          else
            clm(k)%t_soisno(i) = 273.16 - 1.
          endif
        endif
      enddo
    endif
    do i = 1, nlevsoi
      if (lis%o%startcode == 4) then
        if (clm(k)%itypwat == istice) then
          clm(k)%t_soisno(i) = clm(k)%forc_t
        else if (clm(k)%itypwat == istwet) then
          clm(k)%t_soisno(i) = clm(k)%forc_t
        else
          clm(k)%t_soisno(i) = clm(k)%forc_t
        endif
      else
        if (clm(k)%itypwat == istice) then
          clm(k)%t_soisno(i) = clmdrv%clm2_it
        else if (clm(k)%itypwat == istwet) then
          clm(k)%t_soisno(i) = clmdrv%clm2_it
        else
          clm(k)%t_soisno(i) = clmdrv%clm2_it
        endif
      endif
    enddo
    clm(k)%t_grnd = clm(k)%t_soisno(clm(k)%snl+1)
  else !lake

```

```

        if (lis%o%startcode == 4) then
            clm(k)%t_grnd = clm(k)%forc_t
        else
            clm(k)%t_grnd = clmdrv%clm2_it
        endif
    endif
end do

! =====
! Set snow/soil ice and liquid mass
! =====

! volumetric water is set first and liquid content and ice lens are
! then obtained
! NOTE: h2osoi$-$vol, h2osoi$-$liq and h2osoi$-$ice only have valid values
! over soil
do k = 1,numpatch
    clm(k)%h2osoi_vol(           1:nlevsoi) = spval
    clm(k)%h2osoi_liq(-nlevsno+1:nlevsoi) = spval
    clm(k)%h2osoi_ice(-nlevsno+1:nlevsoi) = spval

    if (.not. clm(k)%lakpoi) then !not lake
        if (clm(k)%snl < 0) then !snow
            do i = clm(k)%snl+1, 0
                clm(k)%h2osoi_ice(i) = clm(k)%dz(i)*250.
                clm(k)%h2osoi_liq(i) = 0.
            enddo
        endif
        do i = 1, nlevsoi           !soil layers
            if (clm(k)%t_soisno(i) <= tfrz) then
                if (lis%o%startcode == 4) then
                else
                    clm(k)%h2osoi_ice(i) = clm(k)%dz(i)* &
                        clmdrv%clm2_ism*clm(k)%watsat(i)*denice
                endif
                clm(k)%h2osoi_liq(i) = 0.
                if (clm(k)%itypwat==istwet .or. clm(k)%itypwat==istice) &
                    clm(k)%h2osoi_ice(i)=clm(k)%dz(i)*denice
            else
                if (lis%o%startcode == 4) then
                    print*, 'Not supposed to be called..'
                else
                    clm(k)%h2osoi_liq(i) = clm(k)%dz(i)* &
                        clmdrv%clm2_ism*clm(k)%watsat(i)*denh2o
                endif
                clm(k)%h2osoi_ice(i) = 0.
                if (clm(k)%itypwat==istwet .or. clm(k)%itypwat==istice) &
                    clm(k)%h2osoi_liq(i)=clm(k)%dz(i)*denh2o
            endif
        enddo
    endif
end do

```

```

        endif
    enddo

    do i = 1,nlevsoi
        if (clm(k)%itypwat == istsoil) then
            clm(k)%h2osoi_vol(i) = 0.3_r8
            clm(k)%h2osoi_vol(i) = clm(k)%h2osoi_liq(i)/&
                (clm(k)%dz(i)*denh2o) &
                + clm(k)%h2osoi_ice(i)/(clm(k)%dz(i)*denice)
        else
            clm(k)%h2osoi_vol(i) = 1.0_r8
        endif
        clm(k)%h2osoi_vol(i) = min(clm(k)%h2osoi_vol(i),clm(k)%watsat(i))
    end do
    endif
    end do
end if ! end of arbitrary initialization if-block

! =====
! Remaining variables are initialized by calls to ecosystem dynamics and
! albedo subroutines.
! Note: elai, esai, frac_veg_nosno are computed in Ecosysdyn and needed
! by Fwet and SurfaceAlbedo
! Note: fwet is needed in routine clm_twostream (called by clm_surfalb)
! =====

if ( masterproc ) then
    calday = get_curr_calday()
endif
#if ( defined OPENDAP )
    call MPI_BCAST(calday,1,MPI_REAL,0,MPI_COMM_WORLD,ierr)
#endif
    doalb = .true.

    print*, 'DBG: iniTimeVar -- calling clm2lairread', (',iam,')
    call clm2lairread(lis,tile)

#if (defined DGVM)
    call iniTimeConstDGVM()
#endif

if ( masterproc ) then
    tmp_nstep = get_nstep()
endif
#if ( defined OPENDAP )
    call MPI_BCAST(tmp_nstep,1,MPI_INTEGER,0,MPI_COMM_WORLD,ierr)
#endif
    do k = 1,numpatch

```

```
clm(k)%nstep = tmp_nstep
call EcosystemDyn (clm(k), doalb, .false.)
clm(k)%frac_sno = clm(k)%snowdp/(0.1 + clm(k)%snowdp)
clm(k)%frac_veg_nosno = clm(k)%frac_veg_nosno_alb
if ( clm(k)%itypveg == 12 ) then
    clm(k)%frac_veg_nosno = 0.0
endif
call Fwet(clm(k))
call SurfaceAlbedo (clm(k), calday, eccen, obliqr, lambm0, mvelpp)
end do
return

end subroutine iniTimeVar
```